

(11)Publication number : 10-240548  
(43)Date of publication of application : 11.09.1998

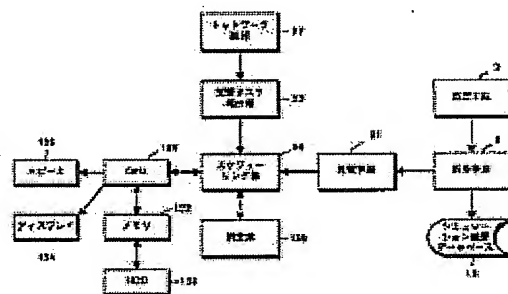
G06F 9/46

(71)Applicant : TOSHIBA CORP

(72)Inventor : FUJIWARA YASUSHI

(57)Abstract:

**SOLUTION:** A specifying means 3 specifies the service quality to be provided by the execution of the task and a sectioning means 6 sections the supplied respective tasks into an essential part to be surely executed so as to satisfy the specified service quality and an option part to be executed when possible. A scheduling part 39 schedules the essential part and schedules the option part only in the case that CPU time remains after the execution of the essential part is ended.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-240548

(43) 公開日 平成10年(1998) 9月11日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 9/46

識別記号

3 4 0

F I

G 0 6 F 9/46

3 4 0 E

審査請求 未請求 請求項の数 8 O L (全 16 頁)

(21) 出願番号 特願平9-47990

(22) 出願日 平成9年(1997) 3月3日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 藤原 靖

神奈川県川崎市幸区柳町70番地 株式会社

東芝柳町工場内

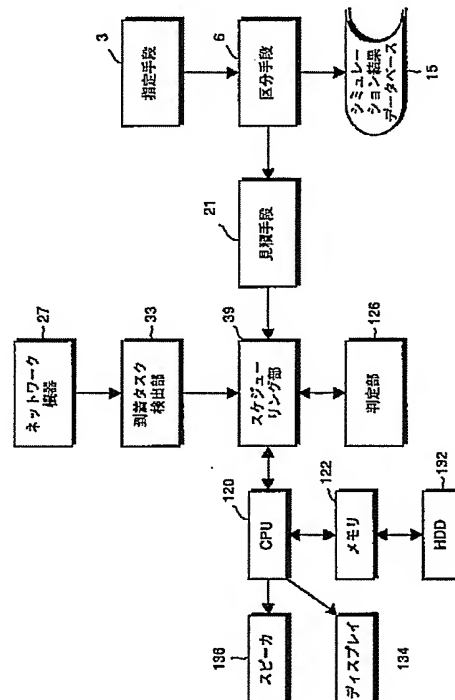
(74) 代理人 弁理士 木内 光春

(54) 【発明の名称】 タスクスケジューリング装置及び方法

(57) 【要約】

【課題】 固定優先順位スケジューリングを用いながら、CPUを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングを実現する。

【解決手段】 指定手段3が、タスクの実行によって提供されるべきサービス品質を指定する。区分手段6が、与えられた各タスクを、指定されたサービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する。スケジューリング部39が、必須部分のスケジューリングを行い、必須部分の実行終了後にCPU時間が余る場合に限って、前記オプション部分をスケジューリングする。



## 【特許請求の範囲】

【請求項1】 与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する区分手段と、各タスクの前記必須部分のスケジューリングと、各タスクの前記オプション部分のスケジューリングを行うスケジューリング手段と、を有することを特徴とするタスクスケジューリング装置。

【請求項2】 前記スケジューリング手段は、必須部分の優先順位をオプション部分の優先順位よりも高く設定し、優先順位の高いタスクにCPUの実行権を順次与え、全ての必須部分の実行が終了した場合に、オプション部分を実行することを特徴とする請求項1記載のタスクスケジューリング装置。

【請求項3】 区分された前記オプション部分をスケジューリングするためのサブスケジューリング手段を有することを特徴とする請求項1又は2記載のタスクスケジューリング装置。

【請求項4】 区分された必須部分の実行所要時間を見積見手段と、見積られた前記実行所要時間に基づいて、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かを判定する判定手段と、を有することを特徴とする請求項1、2又は3記載のタスクスケジューリング装置。

【請求項5】 サービス品質と、タスク中の必須部分と、当該必須部分の実行所要時間との対応関係をあらかじめ記憶させたデータベースを有し、前記区分手段及び前記見積手段は、前記データベースに記憶された対応関係を参照することによって、前記タスクの区分及び前記実行所要時間の見積を行うように構成されたことを特徴とする請求項4記載のタスクスケジューリング装置。

【請求項6】 前記スケジューリング手段及び前記サブスケジューリング手段のうち少なくとも一方は、所定量のCPU時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第2のデータベースを用い、次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにCPU時間を与えた場合の効用を前記第2のデータベースから参照し、当該効用が最大のタスクを次に実行するタスクとして選択するように構成されたことを特徴とする請求項1、2、3、4又は5記載のタスクスケジューリング装置。

【請求項7】 与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分するステップと、各タスクの前記必須部分のスケジューリングと、各タス

クの前記オプション部分のスケジューリングを行うステップと、

を含むことを特徴とするタスクスケジューリング方法。

【請求項8】 前記スケジューリングの際に、所定量のCPU時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第2のデータベースを用い、次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにCPU時間を与えた場合の効用を前記第2のデータベースから参照し、当該効用が最大のタスクを次に実行するタスクとして選択することを特徴とする請求項7記載のタスクスケジューリング方法。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】本発明は、マルチメディア・ネットワーク等の高度な要求に柔軟に対応する組み込みシステム、及びマルチメディアデータ通信システムのための、タスクスケジューリング技術に関わるもので、特に、CPUを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングを実現するものである。

## 【0002】

【従来の技術】コンピュータの急速な普及に伴って、コンピュータを組み込みシステムとして普及する場合が増大している。ここで、組み込みシステムとは、パーソナルコンピュータのようにコンピュータとして独立して使用されるものではなく、家電製品などの電気機器に組み込まれる小規模で機能が制限されたコンピュータシステムである。

【0003】これまでの組み込みシステムは、冷蔵庫や炊飯器のコントローラなど比較的単純なものが多く、8ビット程度のCPUで十分実現可能であった。しかし最近のマルチメディア・ネットワークの多様な普及につれ、生活環境のなかに存在する組み込みシステムへの要求は高度化しつつある。将来的には家電製品もネットワークに接続され、動画や音声などマルチメディアデータをやり取りすることになると思われる。これらの用途では、32ビット以上のCPUの使用が当然のこととなると思われる。高速で強力なCPUの使用は、これまでハードウェアで実現してきた機能をソフトウェア化する可能性をもたらすので、CPU上で多くのタスクを効率的に実行するためのタスクのスケジューリング技術が一層重要となってきた。

【0004】特に、動画や音声などマルチメディアを扱うアプリケーションでは、性能の保証が重要である。動画や音声データは、処理のデッドラインが守れない場合、音声途切れたり、動画がコマ落ちしたりして、品質劣化としてユーザに容易に認識される。このため、これらの分野において、デッドラインを意識したタスクス

ケジューリング、いわゆるリアルタイムスケジューリングが必要である。マルチメディアシステムをこの意味でリアルタイムシステムと考えることも可能だが、マルチメディアシステムに課せられる時間制約は程度問題であって、エレベータ制御における安全管理機能など、これまでの厳密なリアルタイムシステムの時間制約とは、基本的に異なる。これまでのリアルタイムシステムにとって、デッドラインは絶対であった。

【0005】例えば、戦闘機制御システムでは、外部センサーが一定時間毎にデータをサンプリング・処理する。デッドラインを守れないと、撃墜される可能性がある。このように、デッドラインを守ることが最大の優先課題であるようなシステムを、ハードリアルタイムシステムという。これに対して、マルチメディアシステムでは、ある処理をいつまでに必ず行う、という制約は少なく、各種の再生などを一定の品質で行うことの方が重要である。このように厳しいデッドライン制約を持たないリアルタイムシステムは、しばしばソフトリアルタイムシステムと呼ばれる。

【0006】タスクスケジューリングは、オペレーティングシステムの基礎であり、リアルタイムシステムに適用可能なスケジューリングとしては、これまでに多くの手法が提案されている。組み込みシステムに関連するリアルタイムスケジューリングについては、軍事・航空管制などを主なターゲットにした方式がいくつか提案されている。

【0007】複数のタスクをスケジューリングする手法としては、実行可能状態にある各タスクに優先順位（優先度）を割当て、最も高い優先順位を持つタスクに、CPU時間（実行権）を各タスクに与える優先順位スケジューリングが一般的である。この方面で代表的な方式としては、周期的に発生するタスクを周期が短いものほど優先順位を高く設定して、その優先順位に応じたスケジューリングを行うレートモノトニック方式や、デッドラインまでの残り時間が短いタスクほど優先順位を高く設定し、その優先順位に応じたスケジューリングを行うEDF（Earliest Deadline First）方式などがある（“Scheduling algorithms for multi-programming in a hard real-time environment” by C. L. Liu and J. W. Layland, in Journal of the Association for Computing Machinery 20, 1 (January 1973): 40-61”）。

【0008】このように、優先順位に基づくスケジューリング（優先順位スケジューリング）は、優先順位が固定である固定優先順位スケジューリングと、優先順位が動的に変化する動的優先順位スケジューリングに大別さ

れ、レートモノトニック方式は固定優先順位スケジューリングの、EDF方式は動的優先順位スケジューリングの、それぞれ代表的な例となっている。

【0009】固定優先順位スケジューリングでは、タスクに割り当てられた優先順位は、タスクの生成から消滅まで不変である。レートモノトニックスケジューリングは、周期的に実行が必要となるタスクのためのスケジューリング方式で、周期が短いタスクほど優先度を高く設定するものである。例えば、画像データが50msecに一度到達し、音声データが30msecに一度到達するとしよう。この時レートモノトニックスケジューリングでは、周期のより短い音声データを音声化するタスクの優先順位を、画像データを画像化するタスクの優先順位よりも高く設定する。

【0010】図9は、優先順位スケジューリングの一実行例である。ここでは、三つの周期的タスク $\tau_1$ 、 $\tau_2$ 、 $\tau_3$ があり、それぞれが周期15ms、25ms、40msで、周期的に起動される。また、一度の周期で、各タスクは7msだけ実行する。タスクの優先順位は、タスク $\tau_1$ が一番高く、タスク $\tau_2$ が二番目、タスク $\tau_3$ の優先順位が最も低い、と仮定する。この優先順位は、レートモノトニック方式によるものである。

【0011】この場合、時点0で、全てのタスクが起動されるが、優先順位が一番高いタスク $\tau_1$ にCPUでの実行権が与えられ、7ms経過した後、最初の周期を終了する。その後直ちに、タスク $\tau_2$ にCPUでの実行権が与えられ、7ms経過した14msの時点でその最初の周期を終了する。その後、タスク $\tau_3$ にCPUの実行権が渡されるが、15ms時点でタスク $\tau_1$ がその二度目の周期を開始するため、CPUでの実行権はタスク $\tau_3$ からタスク $\tau_1$ へと移される。22ms時点までタスク $\tau_1$ は実行し、そこで二度目の周期を終える。22ms時点では、タスク $\tau_2$ の二度目の周期はまだ始まっていないので、CPUはタスク $\tau_3$ に与えられる。しかし、25ms時点でタスク $\tau_2$ が二度目の周期を開始するため、タスク $\tau_3$ は3msだけ実行した後、タスク $\tau_2$ に実行権を譲り渡す。

【0012】このタスク $\tau_2$ も30ms時点でタスク $\tau_1$ が三度目の周期を開始するため、5msだけ実行した後、タスク $\tau_1$ にCPUを譲り渡す。37ms時点で、タスク $\tau_1$ はその三度目の周期を終え、優先順位が二番目であるタスク $\tau_2$ にCPUの実行権が与えられる。タスク $\tau_2$ は残っていた2msの仕事を行い、39ms時点でその二度目の周期を終える。39msでタスク $\tau_3$ に実行権が戻ってきたが、その最初の周期が終わる40msまでには、残された7-1-3=3ms分の実行は行えず、タスク $\tau_3$ の二度目の周期が開始されてしまい、40msの時点でこのデッドラインを破ってしまう。もっとも、優先順位のいちばん高いタスク $\tau_1$ の実行は確保されており、このことがこの種のスケジューリ

ングの特徴である。その後も、このように優先順位に基づいてタスクがスケジューリングされていく。

【0013】一般に、固定優先順位スケジューリングの方が、実現の容易さ・過負荷時の挙動の予測可能性などから選好される傾向があり、リアルタイムスケジューリング理論ではレートモニタリング方式が主流になっている。マルチメディア関係でも、動画や音声は周期的に発生・到着することから、ネットワークに接続されたマルチメディアファイルサーバに対して、レートモニタリング方式を階層的ラウンドロビン方式と併せ適用しようという提案(USP5528513)などがある。

【0014】

【発明が解決しようとする課題】しかし、従来提唱されているリアルタイムスケジューリング技術は、マルチメディアアプリケーション特有の要求を十分満たしてはいない。まず、マルチメディア対応の組み込みシステムでは、コストの問題から、CPUの能力にはあまり余裕がないのが普通である。しかし、既存のリアルタイムスケジューリング手法は、デッドラインを守るためCPUの能力にはかなり余裕を持たせるのが通常であり、経済的コストの面で不利益を生じる。このため、CPUを効率よく利用するタスクスケジューリング装置が求められていた。

【0015】また、マルチメディアアプリケーションに特徴的なものとして、サービス品質(Quality of Service, QoS)の保証がある。ユーザがマルチメディアアプリケーションにデータの再生要求を出す際は、再生の質についての要求も行う。これをサービス品質の指定というが、マルチメディアアプリケーションは、サービス品質の指定に応じたスケジューリングを行う必要がある(“Resource management in networked multimedia systems” by K. Nahrstedt and R. Steinmetz in IEEE Computer 28, 5 (May 1995): 52-63)。

【0016】これまでのリアルタイムスケジューリング技術では、サービス品質の指定への対応は考えられていなかった。すなわち、レートモニタリング方式では、タスクの優先順位はその周期だけで決まってしまう、ユーザによるサービス品質指定を反映できない。

【0017】一方、優先順位が状況に応じて変化する動的優先順位スケジューリングでは、固定優先順位スケジューリングと比べ柔軟なスケジューリングが可能であり、サービス品質の指定への対応も可能と思われ、サービス品質が重要なマルチメディアシステムでは、動的優先順位に基づくスケジューリング手段(スケジューラ)の使用も提案されている。しかし、動的優先順位スケジューリングには、実現コストが大きい・過負荷時の挙動が予測できないといった課題が残されている。

【0018】優先順位スケジューリングではないが、要求に柔軟に対応するスケジューリング手法としては、imprecise computationと呼ばれるものも存在する(“Imprecise computations” by J. W. S. Liu et al. in Proceedings of the IEEE 82, 1 (January 1994): 83-94)。この手法は、それぞれのタスクを実行が必須な部分とオプションの部分に分割し、必須部分の実行が終了した後オプションの部分を実行する。オプションの部分は、最後まで実行することなく途中で実行を中止することが可能である。imprecise computationでは投入時間とサービス品質はトレードオフの関係にあり、実行時間を長くするほど得られるサービス品質は向上するが、利用できる時間には制限がある。この仮定は、動画再生など多くのアプリケーションに当てはまる。

【0019】しかし、タスクが複数個存在した場合、imprecise computationではオプション部分をスケジュールするためには、非線形計画法やマルコフ決定過程など大掛かりな数学的道具を必要とする。また、imprecise computationでは、スケジューリングを行うのに要する時間が、タスクの個数について指数的に増加するので、組み込みシステムのようにシステム自体は小規模だが、多くのタスクが存在する場合への適用は困難である。

【0020】本発明は、上記のような従来技術の問題点を解決するために提案されたもので、その目的は、優先順位スケジューリングを用いながら、CPUを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングを実現することである。また、本発明の他の目的は、従来のタスクスケジューリング装置を用いたシステムへの導入が容易なタスクスケジューリング装置を提供することである。

【0021】

【課題を解決するための手段】上記の目的を達成するため、請求項1のタスクスケジューリング装置は、与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する区分手段と、各タスクの前記必須部分のスケジューリングと、各タスクの前記オプション部分のスケジューリングを行うスケジューリング手段と、を有することを特徴とする。請求項7のタスクスケジューリング方法は、請求項1の発明を方法の観点から把握したものであって、与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分するステップと、各タスクの前記必須部分のスケジューリングと、各タスクの前記オプション部分のスケジューリングを行うステップと、を含むことを特徴とする。

【0022】請求項1, 7の発明では、サービス品質を指定することができ、タスクは、指定されたサービス品質を満足するのに必須な必須部分と、それ以外のオプション部分に区分される。そして、まず、必須部分のスケジューリングを、レートモニタリング方式のような従来の固定優先順位によって行う。レートモニタリング方式のような従来のリアルタイムスケジューリング理論を用いた部分は厳密なスケジューリング可能性解析が可能であり、このようなスケジューリングを必須部分に適用することによって、サービス品質が確実に保証できる。オプション部分は、必須部分の実行が終了してなおかつ、CPU時間が余る場合に限って実行する。このため、マルチメディア・ネットワークなどにおいて負荷が過剰の場合も、特に指定されたサービス品質に係る必須部分は他の部分より優先実行されるので、サービス品質が保証される。必須部分以外のオプション部分もCPU時間に余裕がある場合は実行されるので、CPUが経済的・効率的に有効活用され、サービス品質が一層向上する。

【0023】請求項2の発明は、請求項1記載のタスクスケジューリング装置において、前記スケジューリング手段は、必須部分の優先順位をオプション部分の優先順位よりも高く設定し、優先順位の高いタスクにCPUの実行権を順次与え、全ての必須部分の実行が終了した場合に、オプション部分を実行することを特徴とする。

【0024】請求項2の発明では、優先順位を可変としておき、必須部分の優先順位をオプション部分よりも高くしておく。この状態で、固定優先順位のスケジューリングを行うことにより、必須部分のみが先に実行される。必須部分の実行が終了した場合は、今度はオプション部分が実行対象となる。このようにすれば、必須部分とオプション部分に、一貫して固定優先順位スケジューリングを適用できる。このため、固定優先順位スケジューリングを用いた従来の組み込みシステムに、本発明を適用しようとする場合も、最小限の追加・変更で済む。

【0025】請求項3の発明は、請求項1又は2記載のタスクスケジューリング装置において、区分された前記オプション部分をスケジューリングするためのサブスケジューリング手段を有することを特徴とする。請求項3の発明では、オプション部分のスケジューリングを専用のサブスケジューラに任せるので、必須部分をスケジューリングする主なスケジューラの負担が軽減され、より経済的なスケジューリングが可能である。

【0026】請求項4の発明は、請求項1, 2又は3記載のタスクスケジューリング装置において、区分された必須部分の実行所要時間を見積る見積手段と、見積られた前記実行所要時間に基づいて、指定されたサービス品質を満足するように各タスクがスケジューリング可能かを判定する判定手段と、を有することを特徴とする。請求項4の発明では、必須部分の実行所要時間を見積られ、さらに、見積られた実行所要時間を用いて、指

定されたサービス品質を満足するように各タスクがスケジューリング可能か否かが判定される。すなわち、指定されたサービス品質を前提として、スケジューリング可能かどうかシステム稼働前に判断できるので、CPUの負担過剰によるシステム障害を事前に回避できる。なお、見積られた実行所要時間は、指定したサービス品質が実現できるかどうかや、オプション部分がどのくらい実行できそうかをユーザなどが判断しようとする場合に、CPU時間の余裕を容易に判断するために用いてもよい。

【0027】請求項5の発明は、請求項4記載のタスクスケジューリング装置において、サービス品質と、タスク中の必須部分と、当該必須部分の実行所要時間との対応関係をあらかじめ記憶させたデータベースを有し、前記区分手段及び前記見積手段は、前記データベースに記憶された対応関係を参照することによって、前記タスクの区分及び前記実行所要時間の見積を行うように構成されたことを特徴とする。請求項5の発明では、あらかじめデータベースに記憶させた情報を用いて、サービス品質から必須部分を定めたり、必須部分の実行所要時間を得ることができるので、処理が正確かつ迅速に行われる。

【0028】請求項6の発明は、請求項1, 2, 3, 4又は5記載のタスクスケジューリング装置において、前記スケジューリング手段及び前記サブスケジューリング手段のうち少なくとも一方は、所定量のCPU時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第2のデータベースを用い、次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにCPU時間を与えた場合の効用を前記第2のデータベースから参照し、当該効用が最大のタスクを次に実行するタスクとして選択するように構成されたことを特徴とする。請求項8の発明は、請求項6の発明を方法の観点から把握したものであって、請求項7記載のタスクスケジューリング方法において、前記スケジューリングの際に、所定量のCPU時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第2のデータベースを用い、次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにCPU時間を与えた場合の効用を前記第2のデータベースから参照し、当該効用が最大のタスクを次に実行するタスクとして選択することを特徴とする。請求項6, 8の発明では、CPU時間を与えた場合の効用（効果）が最大のタスクが実行されるので、システムの性能が最大限に発揮される。

【0029】

【発明の実施の形態】以下、本発明の複数の実施形態について、図面を参照して詳細に説明する。

【0030】1. 第1実施形態

### (1) 構成

図1は、第1実施形態の全体構成を示すブロック図であり、具体的には、本発明によるスケジューリング装置とネットワーク及び周辺装置、典型的にはインターネット端末やビデオオンデマンドなどのマルチメディア機器である。ここでは説明を具体的にするため、ネットワークに接続されたマルチメディアクライアント・サーバシステムを題材にとり、図1のシステムは、マルチメディアサーバ105、複数のマルチメディアクライアント107（請求項1にいうタスクスケジューリング装置に相当するもの）、さらにそれらの間のネットワーク103などからなるものとする。

【0031】マルチメディアサーバ105は、CPU110とメモリ112、タスクのスケジューラ114、スケジューリング可能かどうかの判定部116、大規模なDVDや光ディスクドライブなどからなる記憶装置（ストレージ）101を持ち、ネットワーク103で接続されている。

【0032】マルチメディアクライアント107も、CPU120とメモリ122、タスクのスケジューラ124、スケジューリング可能かどうかの判定部126を持つ。さらに、クライアント107は、マルチメディアデータの再生・表示用に、ディスプレイ134・スピーカ136などのAV機器に接続されている。なお、マルチメディアクライアント107も、ハードディスクドライブ（HDD）など二次の補助記憶132を有し、マルチメディアサーバ105からのデータを格納する場合がある。マルチメディアクライアント107は、ネットワーク103を経由して、マルチメディアサーバ105とつながっている。

【0033】なお、ここでは、動画データ・音声データなどのメディアデータの記録・再生を、コンピュータによって制御・統合するシステムをマルチメディアシステムと呼んでいる。さらにここでは、マルチメディアシステムはスタンドアロンではなく、ネットワークによって結合されていると想定している。このようなシステムにおいて、動画データ・音声データなどの送受信やそれに付随したデータの圧縮・伸長などが典型的なマルチメディアタスクである。サーバーからは毎秒30フレーム送り出される動画データなどがこれにあたる。この場合の周期は33msで、一つのフレームをどの程度のサービス品質で再生するかによって、CPU消費時間は異なる。

【0034】このようなデータを処理するマルチメディアアプリケーションでは、サーバからクライアントまでの間で、サービス品質（QoS: Quality of Service）の保証が要求される。QoSの保証には、サーバやクライアント以外にも、ネットワークノードの役割が大きいが、ここでは、主にサーバ・クライアントにおけるスケジューリングについて説明する。

【0035】次に、図2は、クライアント107の具体的な構成を示す機能ブロック図であり、スケジューラ124の構成要素を具体的に示し、CPU120など、他の要素と共に示したものである。すなわち、クライアント107は、請求項1のタスクスケジューリング装置に対応するもので、タスクの実行によって提供されるべきサービス品質を指定するための指定手段3を有する。また、クライアント107は、与えられた各タスクを、指定されたサービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する区分手段6と、この区分に用いるデータを格納しているシミュレーション結果データベース15（請求項5にいうデータベースに相当するもの）と、を有する。

【0036】また、クライアント107は、区分された必須部分の実行所要時間を見積る見積手段21（請求項4）と、必須部分のスケジューリングを行い、必須部分の実行終了後にCPU時間が余る場合に限り、前記オプション部分をスケジューリングするスケジューリング部39（請求項1にいうスケジューリング手段に相当するもの）と、を有する。また、クライアント107は、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かを判定する判定部126（請求項4にいう判定手段に相当するもの）を有する。

【0037】また、クライアント107は、ネットワーク機器27から到着するタスクを検出する到着タスク検出部33と、スケジューリング部39によるスケジューリングにしたがって、タスクを実行するCPU120と、CPU120とともに情報処理や情報出力を行うためのメモリ122、ハードディスクユニット（HDD）132、スピーカ136及びディスプレイ134を有する。

### 【0038】(2) 作用及び効果

上記のような構成を有する第1実施形態は、次のような作用を有する。すなわち、マルチメディアクライアント107からは、マルチメディアサーバ105に対し、データの送信・停止・再送などの要求が送られる。マルチメディアサーバ105は、マルチメディアクライアント107からの要求により、マルチメディアデータを記憶装置101から検索し、ネットワーク103を通じてクライアント107に送信する。マルチメディアサーバ105からのデータの送信は、多くの場合、データを処理するタスクの周期的な実行を必要とする。

【0039】マルチメディアサーバ105は、多数のマルチメディアクライアント107からの要求に対応する必要がある。これらのマルチメディアクライアント107の要求はユーザのサービス品質指定を反映したサービス品質指定を持つので、マルチメディアサーバ105でも、サービス品質保証を目的としたタスクスケジューリングが必要である。またマルチメディアクライアント107は、ひとつのマルチメディアサーバ105から複数



のデータストリームを受信する場合も、複数のマルチメディアサーバー105から同時にデータを受信する場合もある。マルチメディアクライアント107のスケジューラは、それぞれのデータストリームに対し、ユーザが指定したサービス品質が提供できるよう、タスクスケジューリングを行う。このように、マルチメディアサーバ105・マルチメディアクライアント107は、共にサービス品質を保証するためのスケジューリングを必要とするが、両者で共通のスケジューリング手法が適用できること、構成が簡単であること、本実施形態での例が組み込みシステムであることから、ここではマルチメディアクライアント107のCPUでのタスクスケジューリングについて説明する。

【0040】すなわち、マルチメディアクライアント107のCPUのスケジューラ124に求められるのは、サービス品質指定を充足しつつ過負荷時にも破局的な結果をもたらさないよう、マルチメディアアプリケーションをスケジューリングすることである。スケジューラ124の構成は、レートモノトニックスケジューリングをベースにしたものである。

#### 【0041】(2-1) サービス品質の指定

まず、指定手段3により、サービス品質の指定が行われる。サービス品質の指定は、ユーザがその都度行ってもよいが、タスクやデータの種類によってデフォルト値が自動的に選択されるようにしてもよい。ユーザによるサービス品質指定は、以下により、本スケジューリング装置に反映される。なお、サービス品質の例は、動画における1秒間のフレーム数や、音声におけるサンプリング周波数などである。多くのタスクでは割り当てられたCPU時間によって、サービス品質が定まると考えられる。ユーザの指定したサービス品質を保証するために必要なCPU使用時間を、タスクの中での必須部分の消費するCPU使用時間として設定する。

#### 【0042】(2-2) タスクの区分

サービス品質が指定されると、区分手段6が、タスクを、必須部分とオプション部分に区分する。この区分は、例えば、次のように行うことができる。すなわち、シミュレーション等を行うことによって、タスク中どこまで実行すればどの程度のサービス品質が得られるかのデータを収集してデータベース化し、要求されるサービス品質と必須部分やその実行所要時間との対応関係を判定しておく。この場合、必須部分の実行所要時間は、シミュレーションで求めるだけでなく、プログラムの実行時間を測定する従来の装置を用いて得ることができる。そして、必須部分の量と実行所要時間との対応関係もデータベースに保存しておく。

【0043】このような情報は、シミュレーション結果データベース15(図2)に記憶させておく。ユーザがサービス品質を指定すると、区分手段6がデータベース15を照会してどの部分を必須部分に区分するかを決定

する。通常は、必須部分はタスクの最初の部分であり、オプション部分はタスクのその後の部分である。さらに、見積手段21が、必須部分ごとの実行所要時間を前記データベースから読み出して実行時間を見積ることによって、サービス品質を充足するために必要なCPU時間を計算する。

【0044】なお、判定部126は、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かを判定し、可能と判定された場合に次のスケジューリングが行われる(請求項4)。このように、第1実施形態では、指定されたサービス品質を前提として、スケジューリング可能かどうかをシステムの稼働前に判断できるので、CPUの負担過剰によるシステム障害を事前に回避できる。

#### 【0045】(2-3) スケジューリング

続いて、スケジューリング部39が、各タスクの必須部分について、レートモノトニック方式に従ってスケジューリングを行う。レートモノトニックスケジューリングは、スケジューリング可能性の厳密な判定が可能なので、必ず実行されるべき部分が実際スケジューリング可能であること、言い換えれば指定されたサービス品質が提供できることを厳密に保証できる。レートモノトニック方式では、スケジューリング部39は、必須部分の優先順位をオプション部分の優先順位よりも高く設定し、優先順位の高いタスクにCPUの実行権を順次与え、全ての必須部分の実行が終了した場合に、オプション部分を実行する(請求項2)。

【0046】すなわち、スケジューリング部39は、まず、必須部分の優先順位を高く設定し、オプション部分の優先順位を低く設定する。ここで、図3は、必須部分とオプション部分に区分されたタスクを示す概念図である。ここでは、3つのタスク $\tau_1$ 、 $\tau_2$ 、 $\tau_3$ が存在し、タスク $\tau_1$ は必須部分 $\tau'_1$ 及びオプション部分 $\tau''_1$ に区分され、タスク $\tau_2$ は必須部分 $\tau'_2$ 及びオプション部分 $\tau''_2$ に区分され、タスク $\tau_3$ は必須部分を含まずにオプション部分 $\tau''_3$ のみから構成されるものとする。この場合、実行対象は、図3に示すように、優先順位が高い必須部分310と優先順位が低いオプション部分320の二つに分かれる。必須部分310は具体的には、必須部分 $\tau'_1$ (312)及び $\tau'_2$ (314)で、ユーザのサービス品質指定を保証するために必ず実行する必要がある部分である。オプション部分320は具体的には、オプション部分 $\tau''_1$ (322)、 $\tau''_2$ (324)及び $\tau''_3$ (326)で、必須部分必須部分 $\tau'_1$ (312)及び $\tau'_2$ (314)の実行が終了して余裕がある場合に限り実行対象となる。

【0047】タスク $\tau_1$ (312)、 $\tau_2$ (312)が、ユーザ品質指定を保証するに十分なだけ実行していない間は、必須部分 $\tau'_1$ (312)、と必須部分 $\tau'_2$ (312)を対象としてレートモノトニック方式でス



ケジューリングが行われる。この時点では、矢印316及び318のように、必須部分 $\tau'1$ (312)、と必須部分 $\tau'2$ (312)との間でタスクの切り替えが行われる。しかし、タスク $\tau1$ 、 $\tau2$ について、ユーザ品質指定を保証するに十分なだけ必須部分 $\tau'1$ 及び $\tau'2$ を実行したら、その後、実行対象タスクの切り替えが矢印331、332のように行われ、オプション部分 $\tau''1$ 、 $\tau''2$ 、 $\tau''3$ が、高い優先順位で実行されるタスクがない間だけ、実行される。

【0048】図4は、本実施形態におけるスケジューリングの処理の流れを表したフロー図である。すなわち、スケジューリング部39は、現在実行可能なタスクのうち、最も高い優先順位の与えられたタスクを選びそのタスクにCPUの実行権を与える(ステップ420)。タスクはCPU120で実行されるが(ステップ430)、その間にネットワーク機器27から新しいタスクが届いた場合は(ステップ440)、その到着を到着タスク検出部33が検出し、その新しく到着したタスクも含めて優先順位最高のものを調べ直す(ステップ420)。また、CPUで実行していたタスクが終了したら(ステップ450)、残りのタスクの中で優先順位最高のものを選び直す(ステップ420)。また、高い優先順位で実行される時間が終了したら(ステップ460)、必須部分の優先順位を下げた後(ステップ470)、どのタスクにCPUを与えるか決定するために優先順位を調べ直す(ステップ420)。

【0049】例えば、優先順位として、当初、必須部分 $\tau'1$ に1、必須部分 $\tau'2$ に2、オプション部分 $\tau''1$ に3、オプション部分 $\tau''2$ に4、オプション部分 $\tau''3$ に5を設定したとする。この場合、当初は、優先順位が高い必須部分 $\tau'1$ 又は必須部分 $\tau'2$ のうち、待ち状態ではなく実行可能状態にあるものが交互に実行される。もちろん、必須部分 $\tau'1$ 及び $\tau'2$ の双方が待ち状態の時は、オプション部分 $\tau''1$ などを実行してよい。必須部分のための時間帯が終了すると、必須部分 $\tau'1$ 及び $\tau'2$ はもはやCPUを要求しないので、実行対象がオプション部分 $\tau''1$ 、 $\tau''2$ 、 $\tau''3$ に移る。なお、他の態様として、必須部分 $\tau'1$ 及び $\tau'2$ の優先順位をそれぞれ6と7に引き下げることによって優先順位を逆転させる手法も考えられる。このようにすると、オプション部分 $\tau''1$ 、 $\tau''2$ 、 $\tau''3$ の優先順位の方が、必須部分 $\tau'1$ 及び $\tau'2$ の優先順位よりも高くなるので、実行対象がオプション部分 $\tau''1$ 、 $\tau''2$ 、 $\tau''3$ に移る。

【0050】(2-4)スケジューリングに適した条件なお、第1実施形態によるスケジューリングに適した条件について説明する。各マルチメディアタスク $\tau i$ ( $1 \leq i \leq n$ )が周期的に発生するとし、その周期を $Ti$ で表わす。ユーザのサービス品質指定を満足するために最低限実行する必要がある部分を実行するのに必要なCPU使用時間を、 $Ci$ とする。ここで、 $Ci \leq Ti$ を仮定

するのは当然である。なぜなら、もし $Ci > Ti$ ならば、タスクの次の発生までのCPU時間を全てこのタスクに割り当てても、ユーザが指定するサービス品質は得られず、ユーザのサービス品質指定に達しないからである。タスクの新しいインスタンスが発生した場合は、古いインスタンスを破棄することも当然であり、我々はスケジューラにこの性質を仮定する。

【0051】タスク $\tau i$ の、ユーザのサービス品質指定を満足するため実行する必要がある部分を必須部分 $\tau' i$ で、それ以外の部分をオプション部分 $\tau'' i$ で表わす。以下では説明を簡単にするため、必須部分 $\tau' i$ 、オプション部分 $\tau'' i$ をタスクであるかのように議論する。以下の議論では、タスク自体よりその消費時間を問題にしているので、必須部分 $\tau' i$ 、オプション部分 $\tau'' i$ を導入することで議論が簡単になるからである。

【0052】次の量をユーザの指定するサービス品質に関するCPU消費要求と呼び、 $\mu$ で表わす。

【数1】

$$\mu = \sum_{i=1}^n Ci/Ti$$

この値は、個々のタスクが要求するCPU占有割合の合計であり、個々のタスクのCPU消費要求は、指定されたサービス品質を確保するために必要なCPU使用時間を、タスクの発生周期で除したものである。 $\mu > 1$ の場合は、ユーザのサービス品質指定が実現が不可能であることが、直ちに分かる。さらに、前掲Liu-Laylandの論文でも結果として示されているように、次の関係が成り立つ場合は、各タスク $\tau i$ の一部分必須部分 $\tau' i$ を、それぞれ周期 $Ti$ のタスクとみなし、周期の短いものほど優先順位を高く設定して、優先順位に従ってスケジューリングを行うと、全ての必須部分 $\tau' i$ は次の周期までに実行される。

【数2】

$$\mu \leq n(2^{1/n} - 1) \sim \log 2 = 0.697(n \rightarrow \infty)$$

これは、全てのタスクのCPU使用率の合計が約70%までであれば、すべてのタスク $\tau i$ について、指定のサービス品質が提供される、すなわち、サービス品質指定が必ず実現可能であることを意味する。すなわち、タスク数 $n$ を無限大にすると、 $\mu$ は0.697に収束するが、第1実施形態では $\mu$ を0.697以下とする。これ以上にした場合でも特定の条件の場合には、スケジューリング可能であるが、

【数3】

$$n(2^{1/n} - 1) \leq \mu \leq 1$$

の場合は、組み合わせ的な性質により実現可能か否かが微妙に影響されるため、スケジューリング可能であるための必要十分条件を数値的に与えることはできない。この点、安全サイドにたつて、ユーザ品質指定の実現可能性保証を与えるのは、

【数4】

$$\mu \leq n(2^{1/n} - 1)$$

の場合に限ることが適当と考えられる。一方で、 $\mu$ を上記の範囲以外にする場合を含め、実行が必要な部分がレートモノトニック方式でスケジューリング可能かどうかを判定するための必要十分条件も知られているので、それを利用することも可能である。

【0053】これまで述べてきたのは必須部分 $\tau' i$ が一つ以上実行可能状態の場合であり、実行可能な必須部分 $\tau' i$ が存在しないとき、オプション部分 $\tau' i$ 達をどうスケジューリングするか、については、何も述べていない。このような場合のスケジューリング方式は多数考えられるが、ここでは必須部分 $\tau' i$ の実行が終了した時点で、オプション部分 $\tau' i$ の優先順位を、必須部分 $\tau' i$  ( $1 \leq i \leq n$ ) で最大のものよりもさらに低いものに設定する。そのうえで、スケジューリングは、これまで通りの優先順位スケジューリングに基づいて行われる。この方式の利点は、これまでのシステムをそのままサービス品質制御に転用できることである。この記述から明らかなように、固定優先順位スケジューリング（フラットスケジューリング方式）を応用した第1実施形態は、従来技術とは異なった優れた効果を有するものである。特に、第1実施形態は、従来の固定優先順位スケジューリングに、タスクの優先順位変更機能を追加することによって、わずかな変更で実現できる点で優れている。

【0054】一方で問題となるのは、どのように優先度を変更したらよいかである。組み込みシステムでは、コスト性能比が重要であり、必須部分 $\tau' i$ がCPUを使用していないときでも、できるだけ有効にCPU資源を利用することが望ましい。

【0055】なお、フラットスケジューリング方式の性能は、優先順位決定アルゴリズムに依存する側面があるので、単にやみくもに優先度を下げただけでは、ユーザにとり好ましいサービス品質を提供することは困難である。一例として、タスク数を $n$ として、必須部分を終わらせた優先順位を $n+1$ に設定する、などの単純な方法に限定することは、サービス品質の観点からは必ずしも良い選択ではない。優先順位を決定するアルゴリズムとして種々考えられるが、本来の優先度を考慮したり、タスクの発生周期を考慮するなど、さまざまな手法がありうる。選択肢の例として、タスク数を $n$ 、必須部分に設定された優先度が $i$ であったとき、必須部分の実行を終えたら優先順位を

- ・  $n+1$  に設定する
- ・  $n+i$  に設定する
- ・  $2n+1-i$  に設定する

などが考えられるが、他の様々なバリエーションも検討すべきである。そして、サービス品質の支点からどれが

よい選択であるかは、具体的な適用対象や適用目的に応じて、実験やシミュレーションなどを行い、具体的に決定すべきである。

【0056】(2-5) 第1実施形態の効果

以上説明したように、第1実施形態によれば、固定優先順位スケジューリング手法と優先順位調整機能を利用することで、既存のシステムに対する変更を最小限にとどめつつ、これまでの組み込みシステムでは困難だったマルチメディア・ネットワークアプリケーションにおけるサービス品質保証が可能になる。特に、第1実施形態におけるスケジューリング方式は、リアルタイムスケジューリング理論に根拠を置いており、厳密なサービス品質を保証できる。

【0057】すなわち、第1実施形態では、サービス品質が指定される。タスクは、サービス品質の満足に必須な必須部分と、それ以外のオプション部分に区分される。そして、まず、必須部分のスケジューリングを、レートモノトニック方式のような従来の固定優先順位によって行う。レートモノトニック方式のような従来のリアルタイムスケジューリング理論を用いた部分は厳密なスケジューリング可能性解析が可能であり、このようなスケジューリングを必須部分に適用することによって、サービス品質が確実に保証できる（請求項1, 7）。オプション部分は、必須部分の実行が終了してなおかつ、CPU時間が余る場合に限って実行する。このため、マルチメディア・ネットワークなどにおいて負荷が過剰の場合も、特に指定されたサービス品質に係る必須部分は他の部分より優先実行されるので、サービス品質が保証される。必須部分以外のオプション部分もCPU時間に余裕がある場合は実行されるので、CPUが経済的・効率的に有効活用され、サービス品質が一層向上する。

【0058】また、第1実施形態におけるスケジューリングでは、優先順位を可変としておき、必須部分の優先順位をオプション部分よりも高くしておく。この状態で、固定優先順位のスケジューリングを行うことにより、必須部分のみが先に実行される。必須部分の実行が終了した場合は、今度はオプション部分が実行対象となる。このようにすれば、必須部分とオプション部分に、一貫して固定優先順位スケジューリングを適用できる。このため、固定優先順位スケジューリングを用いた従来の組み込みシステムに、本発明を適用しようとする場合も、最小限の追加・変更で済む（請求項2）。

【0059】また、第1実施形態では、必須部分の実行所要時間が見積られ、さらに、見積られた実行所要時間を用いて、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かが判定される。すなわち、指定されたサービス品質を前提として、スケジューリング可能かどうかシステムが稼働前に判断できるので、CPUの負担過剰によるシステム障害を事前に回避できる（請求項4）。なお、見積られた実行所要時

間は、指定したサービス品質が実現できるかどうかや、オプション部分がどのくらい実行できそうかをユーザなどが判断しようとする場合に、CPU時間の余裕を容易に判断するために用いてもよい。

【0060】また、第1実施形態では、あらかじめデータベース15に記憶させた情報を用いて、サービス品質から必須部分を定めたり、必須部分の実行所要時間を得ることができるので、処理が正確かつ迅速に行われる（請求項5）。

【0061】特に、第1実施形態は通信ネットワークに接続されているので、通信ネットワークを経由して画像や音声などのマルチメディアデータが複数種類送信される場合も、画像や音声などに応じた複数のタスクをスケジューリングすることによって、指定されたサービス品質を保証することができる。

【0062】2. 第2実施形態

#### (1) 構成

第2実施形態のタスクスケジューリング装置は、第1実施形態と略同様の構成を有するが、次の点で第1実施形態と異なるので、第1実施形態との相違点を中心に説明する。すなわち、図5は、第2実施形態のタスクスケジューリング装置の構成を示す機能ブロック図である。この図に示すように、第2実施形態は、第1実施形態と異なり、区分された前記オプション部分をスケジューリングするためのサブスケジューリング部87（請求項3にいうサブスケジューリング手段に相当するもの）と、サブスケジューリング部87がタスクを選択するためのデータを記録したタスク効用データベース93（請求項6、8にいう第2のデータベースに相当するもの）と、を有する。

#### 【0063】(2) 作用及び効果

第1実施形態においては、オプション部分 $\tau^i$ は優先順位を低く設定し、後に必須部分とオプション部分の優先順位を逆転させたが、スケジューラ自体は、一貫して優先順位に従い、優先順位の高い各必須部分 $\tau^j$ と同様の手法で、オプション部分 $\tau^i$ をスケジューリングしていた。このため、オプション部分 $\tau^i$ にどのような優先順位を与えるのが適当かが問題となる。

【0064】第2実施形態では、各オプション部分 $\tau^i$ をもっぱらスケジューリングする機構を導入する。第2実施形態では、各オプション部分 $\tau^i$ は、これまでのレートモニタリング方式によるスケジューラの監督を離れ、異なるスケジューラの監督下に入る。新たなスケジューラとその監督下にある各オプション部分 $\tau^i$ は、本来のスケジューラからはひとつのタスクであるかのように見えるという意味で、このスケジューリング方式は階層的である。

【0065】この新たなスケジューラ（サブスケジューリング部87）を、ここではサブスケジューラと呼ぶ。それに対し、いままで考えてきた本来のスケジューラ

（スケジューリング部39）を、主スケジューラと呼ぶ。サブスケジューラが複数であることも可能であるし、サブスケジューラ自体がさらに階層的な構造を持つ可能性もある。ここでは二階層からなる階層的スケジューリングを述べるが、本発明は三階層以上の階層的スケジューリングに適用することもできる。

【0066】階層的スケジューリング方式の目的は、サービス品質の一層の向上とCPU資源の有効利用である。ユーザの指定した最低限のサービス品質を保証しただけでは、最大の性能を発揮してユーザの期待に応えることはできない。CPUに余裕がある限り、より高いサービス品質が提供されることがユーザにとって望ましい。ユーザが指定した以上のサービス品質に対応する部分をいかにスケジューリングするかも重要な問題になる。ここでは、全体としてのサービス品質を向上させることを目的として、マルチメディアアプリケーションをスケジューリングする階層的スケジューリング方式の一実現方法を述べる。

【0067】図6は、第2実施形態におけるタスクの構成を示す概念図である。この図に示すように、必須部分 $\tau^1$ （602）、必須部分 $\tau^2$ （604）は、ユーザの品質指定を満足するために割り当てられた時間をまだ使いきっていない間、主スケジューラによってレートモニタリング方式による優先順位が割り当てられ、スケジューリングが行われている。図3に示した第1実施形態の場合と異なるのは、ユーザの品質指定を満足するために割り当てられた時間を使いきった後、オプション部分 $\tau^1$ （612）、オプション部分 $\tau^2$ （614）、オプション部分 $\tau^3$ （616）は、主スケジューラによるスケジューリングの対象とはならず、これら3つ合わせたオプション部分群610として、一括してサブスケジューラによりスケジューリングされることである。オプション部分 $\tau^1$ 、オプション部分 $\tau^2$ 、オプション部分 $\tau^3$ は、主スケジューラからは、サブスケジューラによってまとめてスケジューリングされるため、ひとつのタスクのように見え、この意味で第2実施形態におけるスケジューリングは階層的である。

【0068】すなわち、必須部分 $\tau^1$ （602）及び $\tau^2$ （604）間での制御移転（606、608）は主スケジューラによって行われるが、オプション部分 $\tau^1$ （612）、 $\tau^2$ （614）及び $\tau^3$ （616）への制御移転（621、622）はサブスケジューラが行う。

【0069】図7は、第2実施形態におけるタスクスケジューリングの処理手順を示すフローチャートである。第2実施形態におけるスケジューリングでは、まず、主スケジューラのタスクキューが空かどうか、すなわちユーザの品質指定を満足するために割り当てられた時間をまだ使いきっていない実行可能タスクがあるかどうかを調べる（ステップ720）。もしタスクキューが空な

ら、サブスケジューラに制御が移る（ステップ752／後述）。

【0070】もしタスクキューが空でなければ、その中で最も優先順位の高いものにCPUでの実行権を与える（ステップ725）。実行権を与えられたタスクはCPU上で実行されるが（ステップ730）、新しく到着したタスクがあれば（ステップ735）、どのタスクにCPUを与えるかを決定するために優先順位を再び比較する（ステップ725）。新しく到着したタスクがない場合でも、現在のタスクが終了したなら（ステップ740）、主スケジューラのタスクキューが空であるかのチェックに戻る（ステップ720）。

【0071】現在のタスクが終了しない場合でも、もし現在のタスクに割り当てられた高い優先順位で実行する時間が終了したら（ステップ745）、現在のタスクを主スケジューラのタスクキューから外しサブスケジューラのタスクキューに移し（ステップ750）、その後主スケジューラのタスクキューが空かどうかのチェックに戻る（ステップ720）。これらの条件がどれも成立しない限りは、このタスクをCPU上で実行し続ける（ステップ730）。

【0072】ステップ720において、主スケジューラのタスクキューが空の場合、制御はサブスケジューラに移る。この場合、まず、サブスケジューラのタスクキューが空かどうかを調べる（ステップ752）、もし空でない場合は、サブスケジューラは所定の基準に基づいて、タスクキューからCPUを割り当てるタスクを選択する（ステップ755）。選択されたタスクはCPU上で実行されるが（ステップ760）、その間に新しいタスクが到着した場合は（ステップ765）、CPUを取り上げられて、制御は主スケジューラに移され、主スケジューラのタスクキューをチェックする（ステップ720）。

【0073】新しいタスクが到着せず、現在のタスクが終了したら（ステップ770）、サブスケジューラのタスクキューをチェックし（ステップ652）、サブスケジューラの基準にしたがって、どのタスク（オプション部分）何をスケジューリングするかを決定する（ステップ755）。そうでない場合、主スケジューラからサブスケジューラに移されたタスクがあれば（ステップ775）、再びスケジューラのポリシーに従いどのタスクにCPUを与えるかを選択する（ステップ755）。

【0074】そうでない場合、クオンタを使い果たしたら（ステップ780）、このタスクをCPUから外しサブスケジューラの基準に基づいてどのタスクをスケジューリングするかを決定する（ステップ755）。このどれも成立しない限り、タスクはCPU上で実行され続ける（ステップ760）。ステップ752において、サブスケジューラのタスクキューが空である場合は（ステップ753）、新しいタスクが到着したかどうかを調べ

（ステップ785）、もし到着すれば主スケジューラに制御を移し（ステップ725）、到着しない間はこのチェック（ステップ785）を繰り返す。

【0075】なお、サブスケジューラ87は、CPUで実行するタスク（オプション部分）を、次のような基準で選択すればよい。まず、全てのサービス品質は何等かの形で定量化が可能で、異なるタスクのサービス品質が比較できる、との仮定を置く。例えば、動画についてのサービス品質と音声についてのサービス品質が比較できるとする。もちろんこのような数量化は絶対的なものではないが、経済学ではしばしば用いられ、その比較可能な量は効用（utility）と呼ばれているので、ここでもこの用語法を採用する。ユーザの指定するサービス品質は、それぞれのタスクに関する、効用のしきい値と考えられ、それ以下の効用はユーザに受け入れられない。

【0076】マルチメディアアプリケーションでは、CPU使用時間の割り当てを増加させても、サービス品質はそれほど向上しないことがしばしばある。例えば、CPU使用時間を倍にしても、音声再生のサービス品質は倍にならない。本明細書にいう効用に対しても、経済学で言うところの収穫逨減現象が当てはまると思われる。

【0077】収穫逨減現象は、CPU消費時間に対してサービス品質を描いたグラフは、上に凸である、と言い換えられる。この性質は必ずしも成立する必要はないが、以下に述べるタスクの選択基準は、このような場合に最適なものとなっている。また、CPU消費時間と効用の関係は既知であり、データベースもしくは計算式等として、利用可能であると仮定する。

【0078】すなわち、サブスケジューラは、現在実行可能な各オプション部分 $\tau^i$ それぞれに対し、次のクオンタ（固定の時間幅）を割り当てたときの効用の増加量を計算し、増加量が最大になるオプション部分 $\tau^i$ にCPUを割り当てる。なお、このような効用の増加量はあらかじめ計算しておき、その情報をタスク効用データベース93に記憶させておく。

【0079】ここで、図8を参照して実例を説明する。説明を単純にするため、二つのタスク $\tau^1$ と $\tau^2$ があったとし、それぞれの効用と時間との関係が、図8（a）のグラフ810及び図8（b）のグラフ830で与えられているとする。クオンタを $\delta$ とし、効用の変化量を見ると、タスク $\tau^1$ における変化量820の方がタスク $\tau^2$ における変化量840よりも大きい。このため、最初のクオンタはタスク $\tau^1$ に割り当てられる。しかし、次のクオンタについては、タスク $\tau^1$ に割り当てた場合の効用の変化量825よりもタスク $\tau^2$ に割り当てた場合の変化量840の方が大きいので、次のクオンタはタスク $\tau^2$ に割り当てられる。さらにその次のクオンタについても、同様にタスク $\tau^2$ に割り当てられる。このような手法によって、CPU時間を最も効果的に活用する

ことができる。

【0080】この方式は、効用に関して常に最適であるとは限らないが、効用がCPU消費時間の増加に対し非減少かつ上に凸との性質を満たす場合には最適である。もちろん、この方式によるスケジューリング結果は、効用の与え方に大きく依存する。例えば、CPU消費時間に対して定数であるとして、効用を与えることも可能であり、この場合はどのようにスケジュールしても良い。

【0081】ここでは、サブスケジューラの用いるアルゴリズムとして、効用に基づくものを述べたが、それ以外のアルゴリズムを採用することも当然に可能である。例えば、タイムシェアリングシステムでよく用いられるラウンドロビン方式は、タスクに一定のクオンタを割り当て、タスクがそのクオンタを使い果たすと、タスクキューの最後に移され、次のタスクがCPUを割り当てられる、という手法であるが、サブスケジューラの方式として適用できる。

【0082】以上のように、第2実施形態では、オプション部分のスケジューリングを専用のサブスケジューラに任せるので、必須部分をスケジューリングする主なスケジューラの負担が軽減され、より経済的なスケジューリングが可能になるとともに、一層のサービス品質向上が可能になる。また、第2実施形態では、CPU時間を与えた場合の効用（効果）が最大のタスクが実行されるので、システムの性能が最大限に発揮される（請求項6，8）。

### 【0083】3. 他の実施形態

なお、本発明は上記実施形態に限定されるものではないので、次に例示するような他の実施形態をも包含するものである。例えば、上記各実施形態ではもっぱら組み込みシステムを例に議論したが、この手法はマルチメディア通信システムのスケジューリングにも等しく適用できることは明らかである。

【0084】また、タスク管理には次のような手法を適用することができる。例えば、区分された前記オプション部分ごとに生存期間を設定する。タスクのオプション部分がCPUに割り当てられようとする場合には、生存期間が所定の値を超えているか否かを判定する。この結果、生存期間が所定の値を超えているタスクは廃棄する。このように、割り付けられそうになったタスクの生存期間をチェックし、タスクが生存期間を越えていればそれを廃棄することで、タスクの生成と消滅が繰り返されるような場合でも、未処理のタスクが増大し続けないようにできる。

【0085】また、タスクの一部が選択された場合に、当該選択された部分によって、指定されたサービス品質が満足できるか否かを判定する手段を設けてもよい。このようにすれば、スケジューリングの一部を手作業で行ったり、タスクスケジューリング装置のデバッグを行うことが容易になる。なお、このような判定は、例

えば、サービス品質とその実現に必要なタスクの部分との対応関係をあらかじめ調べてデータとして記憶させておき、記憶されているタスクの部分が、選択された部分に含まれているかどうかを調べることによって、実現することができる。

【0086】なお、本発明のタスクスケジューリング装置及びタスクスケジューリング方法は、コンピュータプログラムを用いて実現されることが一般的と考えられるが、そのようなコンピュータプログラムを記録したフロッピーディスクなどの記録媒体も本発明の一態様である。

### 【0087】

【発明の効果】以上説明したように、本発明によれば、CPUを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングが実現される。

### 【図面の簡単な説明】

【図1】本発明の第1実施形態におけるマルチメディアクライアント・サーバシステム全体の概略構成を示すブロック図。

【図2】本発明の第1実施形態におけるクライアントの具体的構成を示す機能ブロック図。

【図3】本発明の第1実施形態におけるタスクを示す概念図。

【図4】本発明の第1実施形態におけるタスクスケジューリングの処理手順を示すフローチャート。

【図5】本発明の第2実施形態におけるクライアントの具体的構成を示す機能ブロック図。

【図6】本発明の第2実施形態におけるタスクを示す概念図。

【図7】本発明の第2実施形態におけるタスクスケジューリングの処理手順を示すフローチャート。

【図8】本発明の第2実施形態における投入時間（クオンタ）とそれによる効用との関係を示すグラフ。

【図9】レートモニタリングスケジューリングの例を示すタイムチャート。

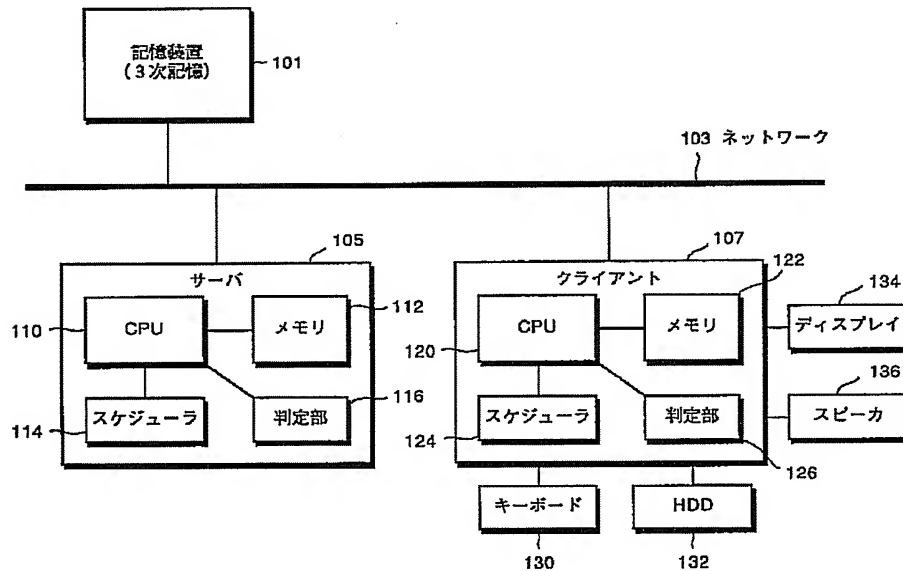
### 【符号の説明】

- 3…指定手段
- 6…区分手段
- 15，93…データベース
- 21…見積手段
- 27…ネットワーク機器
- 33…到着タスク検出部
- 39…スケジューリング部
- 87…サブスケジューリング部
- 101…記憶装置
- 103…ネットワーク
- 105…サーバ
- 107…クライアント
- 110，120…CPU
- 112，122…メモリ

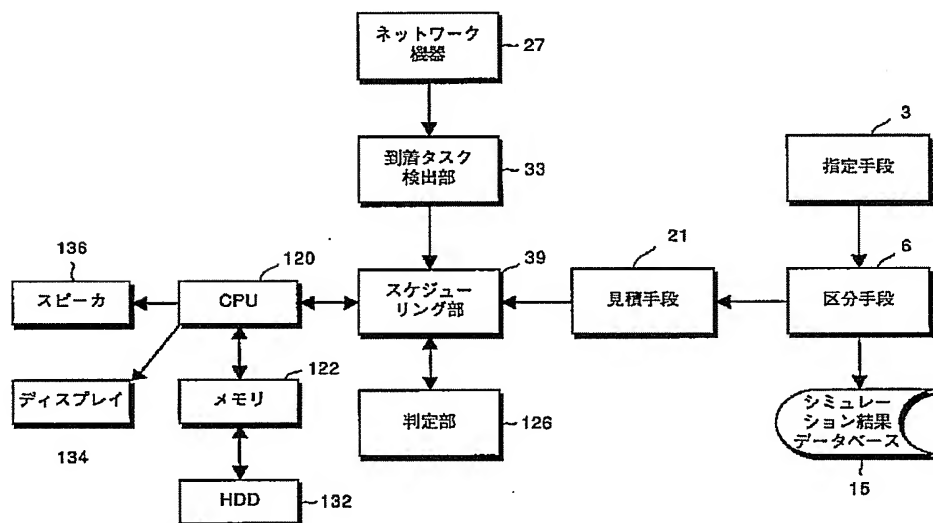
114, 124…スケジューラ  
 116, 126…判定部  
 130…キーボード  
 132…ハードディスクドライブ  
 134…ディスプレイ  
 136…スピーカ

310, 602, 603…必須部分  
 320, 610…オプション部分  
 810, 830…グラフ  
 820, 825, 940, 945…変化量  
 STEP…手順の各ステップ

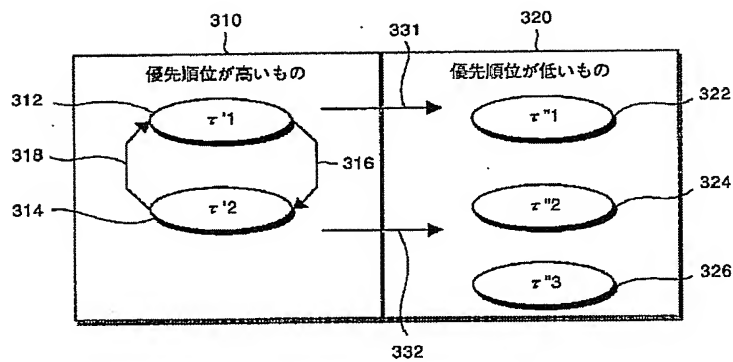
【図1】



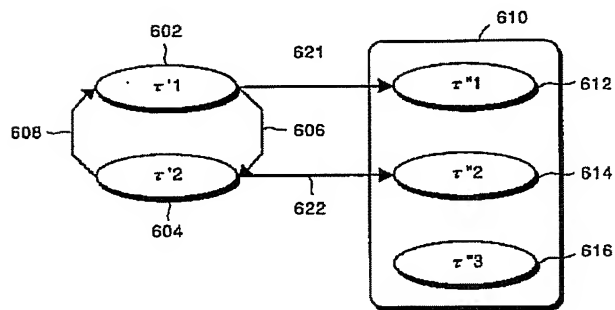
【図2】



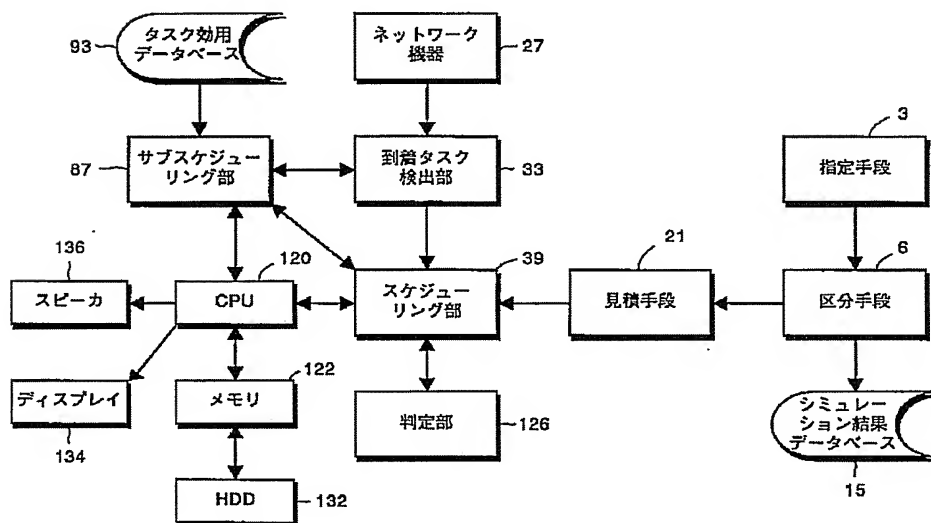
【図3】



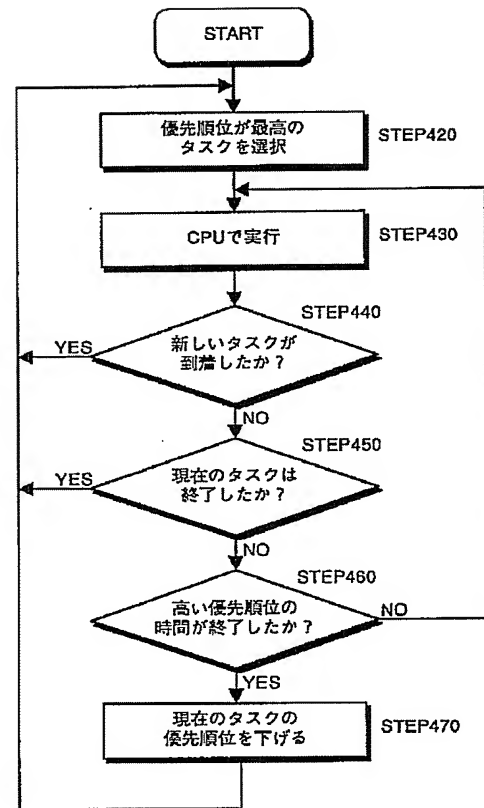
【図6】



【図5】

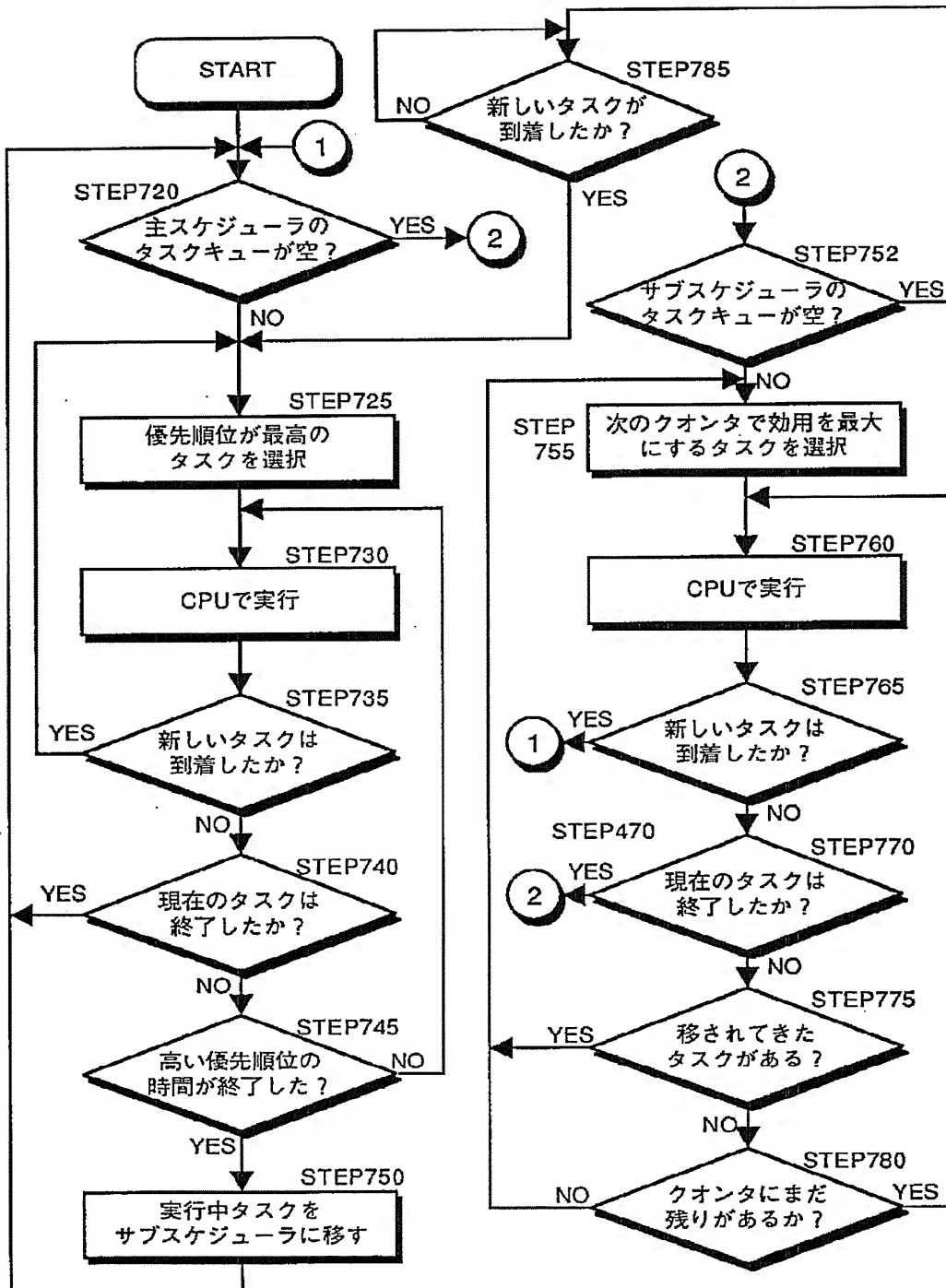


【図4】

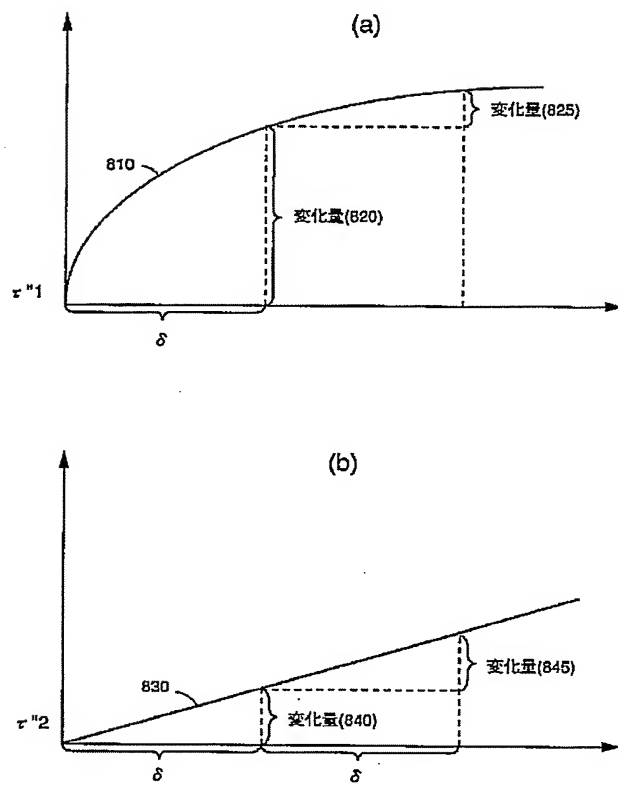




【図7】



【図8】



【図9】

